JOHN WILEY & SONS
605 Third Avenue, New York, N.Y. 10158
New York • Chichester • Brisbane • Toronto • Singapore

FILE INPUT/OUTPUT STATEMENTS (continued)

OPEN—Opens a channel for input and/or output to the designated devices.

format: OPEN *file#* [*device#, address, string*]

example: *line* OPEN 1,1,0, "DATA"

where: *file#* ranges from 0 to 255 and relates the OPEN, CLOSE, CMD, GET#, INPUT# and PRINT# statements to each other.

device# specifies the peripheral device.

address is a code that tells each device what operation to perform.

Note: The string can be used for the filename with cassette operations or can be a filename or control information when used with disk.

when used with disk, string is [,"*filename*[,*type*][,*mode*"]]

where: *type* is { PRG for program file / SEQ for sequential file / USR for user file } default type is sequential

where: *mode* is { R to read sequential file / W to write sequential file }

PRINT#—Sends the contents of the variables in the list to the device previously OPENed.

format: PRINT# *file#,var* { , ; } [*var* ...] { , ; }

example: *line* PRINT#1, "ANSWER IS"; X

Note: PRINT# is one keyword with no space after PRINT. The characters ?# may not be used as an abbreviation for PRINT#.

INPUT/OUTPUT CONTROL PARAMETERS

File# can be any number from 1 to 255 and is the same number that will be used in the INPUT#, GET# and PRINT# statements to work with this device. Since file# exceeding 127 were designed for other uses, numbers 1 to 127 should normally be used. *Device#* specifies the physical address of the device. *Address* specifies the operation to be performed based upon the device, and *string* specifies file name or control information.

DEVICE ALLOCATION TABLE

DEVICE#	DEVICE
0	Keyboard
1	Cassette Deck
2	RS232 Device
3	Screen
4	Printer
5	Printer
6–7	Other serial bus devices
8–11	Disk Drive
9–31	Other serial bus devices

address specifies the operation to be performed based upon the device.

string specifies file name or control information.

ADDRESS CODE OPERATION TABLE

DEVICE	ADDRESS CODE	OPERATION	STRING
Cassette	0	read tape file	filename
	1	write tape file	filename
	2	write tape file and place EOT marker at end	filename
Disk	0	LOAD	file type, read
	1	SAVE	write command
	2–14	open data channel	drive#, filename
	15	command channel	
Screen	0,1	no effect	
Printer	0	uppercase/graphics	text is printed
	1–7	special features (refer to printer manual)	

Designed by Werner Graphics
ISBN 0 471-88240-2

PRINTER COMMAND TABLE (for Commodore Printer)

PRINT COMMAND	OPERATION	PRINT COMMAND	OPERATION
CHR$(10)	Line feed after printing	CHR$(26)	Repeat graphic select command
CHR$(13)	Carriage return	CHR$(145)	Cursor up (upper-case) mode
CHR$(8)	Graphic mode command	CHR$(17)	Cursor down (upper/lowercase) mode
CHR$(14)	Double width characters	CHR$(18)	Reverse field on command
CHR$(15)	Standard character mode	CHR$(146)	Reverse field off command
CHR$(16)	Tab to position in next 2 characters		
CHR$(27)	Prefix to CHR$(16) to specify a dot position by dot address		

CONTROL PARAMETER UTILIZATION EXAMPLES

OPEN 1,0	Opens a channel to read from the keyboard.
OPEN 1,1,0,"DATA"	Opens a channel to read the file named DATA from the cassette tape.
OPEN 1,4,7,"1/15/83"	Opens a channel to send upper/lowercase to the printer and prints the string 1/15/83.
OPEN 4,4:CMD4:LIST	Lists the program in memory on the printer.

NOTE: The string at the end of an OPEN statement is sent to the printer or screen as if a PRINT# statement were used with that device. When the OPEN statement references a cassette deck, it is used for the filename while its use with a disk can be as a filename or for sending control information to the disk.

BASIC FUNCTIONS

FUNCTION FORMAT AND DESCRIPTION

ABS(*exprnm*)—Returns the absolute value of a number

ASC(*expr$*)—Returns the ASC II code number for the first character of the specified string.

ATN(*exprnm*)—Returns the arctangent as an angle of *exprnm* radians.

CHR$(*exprint*)—Returns the character (string value) of the specified ASC II code.

COS(*exprnm*)—Returns the cosine of an angle of *exprnm* radians.

EXP(*exprnm*)—Returns the base of the natural logarithm (e) raised to the specified power.

FRE(*exprnm*)—Returns the number of bytes in memory not being used by BASIC. If the results of FRE are negative, add 65536 to the FRE number to obtain the number of bytes available in memory.

INT(*exprnm*)—Returns the integer position of a number or expression.

LEFT$(*expr$,exprnm*)—Returns the leftmost *exprnm* characters of the string *expr$*.

LEN(*expr$*)—Returns the length of the specified string.

LOG(*exprnm*)—Returns the natural logarithm of the specified number.

MID$(*expr$,exprnm,[,exprnm,]*)—Returns *exprnm*, characters from *expr$*, commencing with character *exprnm*,.

PEEK(*memadr*)—Returns the decimal value of a specified memory location.

POS(*exprnm*)—Returns the current cursor position.

RIGHT$(*expr$,exprnm*)—Returns the rightmost *exprnm* characters of the string *expr$*.

RND(*exprnm*)—Returns a random number between 0 and 1 if exprnm is positive. If *exprnm* is zero, returns, a "randomized" random number. If *exprnm* is negative, returns a preset random number.

SGN(*exprnm*)—Returns + 1 if *exprnm* is positive. – 1 if negative and 0 if its value is zero.

SIN(*exprnm*)—Returns the sine of an angle of *exprnm* radians.

SPC(*exprnm*)—Used with the PRINT statement to print blanks and moves the cursor *exprnm* positions to the right.

SQR(*exprnm*)—Returns the square root.

STATUS / ST—Returns the Commodore 64's status corresponding to the last I/0 operation.

STR$(*exprnm*)—Converts a numeric value to a string.

TAB(*exprnm*)—Used with the PRINT statement to move the cursor to the specified position.

TAN(*exprnm*)—Returns the tangent of the angle of *exprnm* radians.

TIME / TI—Returns the value of the interval timer in one-tenth seconds.

TIME$ / TI$—Reads the internal interval timer and returns a string of 6 characters in hours, minutes and seconds.

USR(*exprnm*)—Calls the user's assembly language subroutine whose starting address is stored in locations 1 and 2.

VAL(*expr$*)—Returns the numeric value of a string.

SOUND AND MUSIC TABLES

COMMAND		VALUES OF *X*	FUNCTION	DESCRIPTION
POKE 54296	,*X*	0 to 15	volume	sets volume
POKE 54277 / 54284 / 54291	,*X*	see attack/decay table	attack/decay	sets voice 1,2,3 rise and fall times
POKE 54278 / 54285 / 54292	,*X*	see sustain/release table	sustain/release	prolongs voice 1,2,3 note
POKE 54273 / 54280 / 54287	,*X*	see musical note table	high frequency	sets voice 1,2,3 high frequency note
POKE 54272 / 54279 / 54286	,*X*	see musical note table	low frequency	sets voice 1,2,3 low frequency note
POKE 54276 / 54283 / 54290	,*X*	see waveform table	waveform	defines voice 1,2,3 waveform

ATTACK/DECAY RATE SETTINGS

HIGH ATTACK	MEDIUM ATTACK	LOW ATTACK	LOWEST ATTACK	HIGH DECAY	MEDIUM DECAY	LOW DECAY	LOWEST DECAY
128	64	32	16	8	4	2	1

SUSTAIN/RELEASE RATE SETTINGS

HIGH SUSTAIN	MEDIUM SUSTAIN	LOW SUSTAIN	LOWEST SUSTAIN	HIGH RELEASE	MEDIUM RELEASE	LOW RELEASE	LOWEST RELEASE
128	64	32	16	8	4	2	1

WAVEFORM CONTROL SETTINGS

TRIANGLE	SAWTOOTH	PULSE	NOISE
17	33	65	129

MUSICAL NOTE TABLE

FREQUENCY	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
HIGH	34	36	38	40	43	45	48	51	54	57	61	64	68	72
LOW	75	85	126	200	52	198	127	97	111	172	126	188	149	169

SCREEN CODE

Code sets switched by holding down the SHIFT and C= keys simultaneously.

Codes from 128–255 are reversed images of codes 0–127.

POKE 36869,240 sets character set to uppercase POKE 36869,242 sets character set to lowercase

COLOR CODE TABLE

Code	Color	Code	Color	Code	Color	Code	Color
0	Black	4	Purple	8	Orange	12	Gray 2
1	White	5	Green	9	Brown	13	Light Green
2	Red	6	Blue	10	Light Red	14	Light Blue
3	Cyan	7	Yellow	11	Gray 1	15	Gray 3

SCREEN AND BORDER COLOR COMBINATIONS

POKE 53280, *varnm* changes border color POKE 53281, *varnm* changes screen color

Values of the numeric variable (*varnm*) must be between 0 and 15 and represent the selected color code.

DISPLAY EXAMPLE

Statement	Operational Result
POKE 53280,0	sets border to black
POKE 53281,2	sets background color to red
POKE 36869,240	sets character set to uppercase (set 1)
POKE 1024,36	places $ in upper left corner of screen
POKE 55796,7	colors the $ yellow

COMMODORE 64 BASIC

QUICK REFERENCE GUIDE

by Gilbert Held

WILEY QUICK REFERENCE GUIDES 2.95

REFERENCE GUIDE NOTATIONS AND FORMAT CONVENTIONS

A standard scheme for presenting the general format of BASIC language statements is employed in this reference guide. The capitalization, punctuation and other conventions are listed below:

[ ] Brackets indicate that the enclosed items are optional. Brackets do not appear in the actual statements.

{ } Braces indicate that a choice of one of the enclosed items is to be made. Braces do not appear in the actual statements.

Ellipses indicate that the preceding item may be repeated. Ellipses do not appear in the actual statements.

Italics Italics indicate generic terms. The programmer must supply the actual value or wording required. See Generic Terms and Abbreviations.

Line number A line number is implied for all BASIC language statements in program mode.

Punctuation All punctuation characters, including commas, semicolons, colons, quotation marks and parentheses, must appear as indicated.

UPPERCASE Uppercase letters and words must appear exactly as indicated.

BASIC PROGRAMMING MODES

DIRECT—Statement(s) entered without a line number will be immediately executed by BASIC.

PROGRAM—Statement(s) entered with line numbers will be executed by the RUN command.

BASIC STATEMENT FORMATS

Maximum line length is 80 characters on 2 physical lines of 40 characters per line.

Multiple statements permitted on a line using the colon (:) as a statement separator.

Direct Mode format: *statement* [:*statement* . . .]

Program Mode format: *line statement* [:*statement* . . .]

GENERIC TERMS, ABBREVIATIONS AND DEFINITIONS

arg—Argument.

array—A set of variables that has the same name and that is distinguished by a number known as the subscript written in parenthesis after the name. An array can have as many values as there are elements, with each element of the array having a separate value.

command#—A number specific to a device that selects a specific channel or activity within a device.

const—Any string or numeric constant.

constant—A value that does not change.

device—A Commodore 64 component, such as the keyboard, or an attachment, such as the screen, tape recorder, printer or disk drive.

device#—A number that specifies a given device.

expr—Any valid Commodore 64 expression.

expr$—Any valid Commodore 64 string constant, variable or expression.

exprint—An integer expression.

exprnm—Any numeric constant, variable or expression.

filename—A cassette or disk file name.

file#—A number from 1 to 255 used in the CLOSE, CMD, OPEN, INPUT#, GET# and PRINT# statements to work with a device.

floating point—Number with a decimal point.

format—The structure of a BASIC command or statement.

integer—A whole number ranging between −32768 and +32767.

line—A BASIC program line number.

memadr—The memory address referenced by a numeric expression, variable or constant.

print zone—The Commodore 64's display is organized into 4 areas of 10 columns each, each area known as a print zone.

program name—A name consisting of up to 16 characters that defines the name of a file containing a program on cassette or disk.

sprite—A high resolution programmable object contained in a 24 by 21 position dot grid.

statement—A BASIC language statement.

string—One or more characters enclosed in double quotation marks.

sub—Subscript.

var—Numeric, string or integer variable.

varnm—A numeric variable name.

var$—A string variable.

DISPLAY CONTROL

SCREEN EDITING permits cursor to move around the screen and allows you to make changes to program lines.

CLR HOME When unshifted, moves the cursor to the upper left corner of the screen. When used with the SHIFT or C= key held down, clears the screen and moves cursor to upper left corner of the screen.

DISPLAY CONTROL (continued)

INST DEL Deletes character to the left of the cursor. Anything else on the line shifted one space to left. When used with the SHIFT key a space is inserted at the cursor's position and everything on the line to the right of the cursor is moved one space to the right.

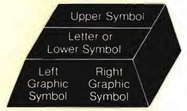
CRSR ↓ Moves cursor down one line when unshifted. Moves cursor up one line if the SHIFT or C= key held down.

CRSR → Causes cursor to move to the right if unshifted. Causes the cursor to move to the left if SHIFT key held down.

RETURN Causes a command or statement to be entered. If SHIFT key held down causes the cursor to move to the next line.

SPACE Causes a blank space to be generated on the screen and the cursor to move one space to the right.

KEY UTILIZATION



Press key to display letter or lower symbol.

Press shift and key to display upper symbol.

Press C= and key to display left graphic symbol.

Press SHIFT and key to display right graphic symbol.

CHARACTER SET SELECTION

Press C= and SHIFT to switch between character set 1 and character set 2.

Character set 1 is normal uppercase letters, the digits 0 through 9 and all graphic characters.

Character set 2 includes both uppercase and lowercase letters, the digits 0 through 9 plus some graphic characters.

PRINTING CONTROL

The use of quote marks or the INS key permits cursor controls, color controls, and function keys to be entered as "programmed" reverse characters.

Color Control—Press CTRL key and any one of 8 color keys.

Reverse Character—Press CTRL key and RVS ON key to commence reverse video.

Press CTRL key and RVS OFF or a PRINT RETURN to end reverse video

VARIABLE NAMING CONVENTIONS

Name format: **F[ST]**

where: **F**—represents the first character which must be alphabetic.

S—is an optional second character that can be alphabetic or numeric.

T—identifies the type of variable as follows:

% for integer.

$ for string.

If Type is omitted BASIC assumes variable is a floating point number.

Note: Variable name length can be up to 255 characters; however, only first 2 characters count.

Examples:

A—represents floating point variable.

A%—represents integer variable.

A$—represents string variable.

CAUTION: AL and ALPHA will be treated as the same name since only the first two characters count in variable names. In addition, when variable names contain two or more alphabetic characters, the user should be careful that there is no conflict with BASIC keywords, such as IF, TO or ST.

BASIC OPERATORS

Operation	Operator	Example
ARITHMETIC		
Exponentiation	↑	A↑B
Unary Minus	−	− A
Multiplication	*	A*B
Division	/	A/B
Addition	+	A + B
Subtraction	−	A − B
RELATIONAL		
Equal	=	A = B
Not equal to	< >	A < > B

Operation	Operator	Example
RELATIONAL (continued)		
Less than	<	A < B
Greater than	>	A > B
Less than or equal to	< =	A < = B
Greater than or equal to	> =	A > = B
BOOLEAN		
Logical complement	NOT	NOT A
Logical AND	AND	A AND B
Logical OR	OR	A OR B

SYSTEM COMMANDS

These commands result in the computer performing an operation at the system level. The commands are normally entered without a line number; however, most commands can also be used in a program by prefixing the command with a line number.

CONT—Restarts the execution of a program previously stopped by the pressing of the STOP key or the execution of a STOP or END statement within a program. Program will restart at the exact place it previously terminated.

format: CONT

LIST—Causes the entire program or the indicated program lines to be displayed.

format: LIST[*line*$_1$] − [*line*$_2$]

LOAD—Causes a program from cassette tape or disk to be transferred into the Commodore 64's memory, erasing any BASIC program previously entered into the computer. If no filename is specified the first program encountered on tape will be loaded.

format: LOAD["*filename*", *device#*, *address*]

Note: Unspecified device# causes program to load from cassette. Unspecified address causes the program to LOAD starting at memory location 2048. If a secondary address of 1 is used, program will LOAD at the memory location from which it was saved.

NEW—Causes the current program to be erased from memory so a new program can be entered from the keyboard.

format: NEW

RUN—Causes the program currently in memory to be executed beginning at its lowest numbered line or at the specified line number.

format: RUN[*line*]

SAVE Causes the program currently in memory to be saved on tape or disk. The program SAVEd will remain in the Commodore 64's memory after the save operation.

format: SAVE ["*filename, device#, address*]

Note: If device# not specified the cassette will be used.

VERIFY Checks the program on tape or disk against the program in the Commodore 64's memory.

format: VERIFY ["*filename*", *device#*]

Note: If device# not specified the cassette will be used.

BASIC LANGUAGE STATEMENTS

BRANCHING

GOSUB—Results in a branch to the indicated line number. A RETURN statement causes a branch back to the instruction following the GOSUB.

format: GOSUB *line*

example: *line* GOSUB 500

GOTO—Causes an unconditional branch to the indicated line number.

format: { GOTO *line* / GO TO *line* }

example: *line* GOTO 500

IF-THEN—Causes the branch or the execution of a statement to occur if the indicated expression is true.

format: IF expr { GOTO *line* / THEN *line* / THEN *statement* }

example: *line* IF X > 4 THEN X = 0:M = M + 1

Note: On IF expr THEN statement, if the expression is false, the entire remainder of the line is not executed.

ON-GOSUB—Causes a conditional subroutine call based upon the current or computed value of the expression. The computed value must be in the range 0 to 255. If the computed value does not have a corresponding line number given, no GOSUB will be performed.

format: ON *exprnm* GOSUB *line*[,*line* . . .]

example: *line* ON X GOSUB 100,200,300

Note: The value of the numeric expression must be in the range 0 to 255. If the value is 0 or exceeds the number of line numbers in the list, the statement will be ignored.

ON-GOTO—Causes a conditional branch based upon the current or computed value of the expression. The computed value must be in the range 0 to 255.

If the computed value does not have a corresponding line number given, no GOTO will be performed.

format: ON exprnm GOTO *line*[,*line* . . .]

example: *line* ON X GOTO 100,200,300

Note: The value of the numeric expression must be in the range 0 to 255. If the value is 0 or exceeds the number of line numbers in the list, the statement will be ignored.

RETURN—Results in a program branch to the statement immediately following the most recently executed GOSUB or ON-GOSUB statement.

format: RETURN

example: *line* RETURN

MEMORY REFERENCE

CLR—Initializes all numeric variables and array elements to zero, assigns a null value to all strings, un-DIMensions all arrays and RESTOREs the DATA pointer back to the beginning.

format: CLR

example: *line* CLR

POKE Places the specified value into the designated memory address.

format: POKE *memadr, exprnm*

where: 0 ≤ *memadr* ≤ 65536
0 ≤ *exprnm* ≤ 255

Example: *line* POKE 1666,32

PROCESSING STATEMENTS

DATA—Creates a list of values to be assigned to variables through the use of a READ statement.

format: DATA *constant*[,*constant* . . .]

example: *line* DATA 1,3,5,"JOHN"

DEF FN—Statement that permits special functions to be defined.

format: DEF FN *letter*[*letter*](*arg*) = *exprnm*

example: *line* DEF FNA(X) = 3*X + 5

DIM—Reserves space in memory for an array or matrix of variables.

format: DIM *var*(*sub*)[,*var*(*sub*) . . .]

example: *line* DIM A(20),B(12),C(12,2)

END—Terminates an executing program and generates the message: READY.

format: END

example: *line* END

FOR—Initiates a loop that repeats execution of all instructions bounded by the corresponding NEXT statement until the automatically incremented variable attains the value *exprnm*$_2$. If STEP clause omitted an increment of +1 is used.

format: FOR *varnm* = *exprnm*$_1$ TO *exprnm*$_2$[STEP *exprnm*$_3$]

example: *line* FOR I = 2 TO 20 STEP 2

GET—Statement that receives one character at a time from the keyboard and assigns it to the specified variable.

format: GET *var$*[,*var$* . . .]

example: *line* GET X$

INPUT—Optionally displays a prompt message and then accepts input data, assigning values to the variables listed.

format: INPUT ["*prompt message*";]*var* [,*var* . . .]

example: *line* INPUT "ENTER NUMBER";X

Note: If no input is entered by the user, variables retain their previous values.

LET—Assigns a value to the specified variable.

format: [LET]*var* = *expr*

example: *line* LET A = B + C

NEXT—Defines the limit of a loop initiated by a FOR statement.

format: NEXT[*varnm*, . . .]

example: *line* NEXT I

PRINT—Outputs values to the display.

format: PRINT[TAB(*exprnm*)]*var* { ; / , } [*var* . . .] { ; / , }

where: TAB moves PRINT position to the column specified.

[,] COMMA moves the beginning of the next item to be displayed to the next print zone on the present line or position 0 on the next line

[;] SEMICOLON continues display immediately after previous output displayed.

example: *line* PRINT "SALES = ";X

Note: PRINT may be abbreviated as ?.

READ—Assigns values from DATA statements to variables in the READ statements.

format: READ *var*[,*var* . . .]

example: READ X,Y,P$

REM—Nonexecutable statement that permits remarks to be placed in a program.

format: REM remark

example: *line* REM OUTPUT RESULTS

Note: REM statements are not terminated by a colon. They continue to the end of the line.

RESTORE—Causes the next READ statement values to be assigned from the first DATA statement in the program.

format: RESTORE

example: *line* RESTORE

STOP—Causes the program to halt execution and display the message: BREAK IN LINE XXXX.

format: STOP

example: *line* STOP

SYS—Calls a machine language program located at the specified address.

format: SYS *exprnm*

example: *line* SYS 64802

WAIT—Halts a program until a specified memory location attains a defined value.

format: WAIT *memadr*, *exprnm*$_1$ [,*exprnm*$_2$]

example: *line* WAIT 36868,144,16

FILE INPUT/OUTPUT STATEMENTS

CLOSE—Causes the file that was started in an OPEN statement to close.

format: CLOSE *file#*

example: *line* CLOSE 1

CMD—Changes the normal output device of the Commodore 64 from the screen to the specified file. This statement permits data and listings to be sent to such devices as the printer, tape drive or disk drive. The string, where specified, is sent to the file.

format: CMD *file#*[,*string*]

example: *line* CMD 1

GET#—Causes data to be received one byte at a time from any OPENed device.

format: GET# *file#*,*var*[,*var* . . .]

example: *line* GET# 1,X$

INPUT#—Retrieves data from the designated file number and assigns them to the specified variables.

format: INPUT# *file#*,*var*[,*var* . . .]

example: *line* INPUT# 1,X$